Experiment 4: Analog Input and Output

Table of Contents

1. Objectives

- Introduce the `PINSELx` registers
- Using the *Analog-to-Digital Converter (ADC)* to read analog input
- Using the *Digital-to-Analog Converter (DAC)* to write analog output

2. Parts List

- LPC1768 mbed board
- USB A-Type to Mini-B cable
- Breadboard
- Light sensor and/or potentiometer
- Seven-segment display or set of LEDs
- 330-Ohm Resistors
- Jumper wires

3. Background

Many microcontrollers have pins that can be used for *analog input*. Because the microcontroller processes digital data only, analog input must be converted to digital data. An analog-to-digital converter (ADC) is an I/O circuit often integrated into microcontrollers to allow directly connecting external analog devices, such as sensors. The ADC would convert the sensor voltage into a digital value by transforming it into a binary code with a specific number of bits.

Although not critical to conducting this experiment, it would be useful to review the three steps involved in analog-to-digital conversion: sampling, quantization, and bit encoding (COE 241).

3.1. Using LPC1768 Peripherals

The LPC1768 includes an integrated ADC peripheral device. In general, using any peripheral device involves three main issues:

1. Powering up the peripheral
2. Configuring the peripheral clock
3. Configuring pin functions

3.1.1. Power Up

All microcontroller peripherals must be powered up before they can be used. This was not a concern in earlier experiments because we were using peripherals that are powered up by default.

Powering peripherals up and down is controlled through the *Power Control for Peripherals Register* (`PCONP`).

By referring to table 46 in Chapter 4 of the LPC176x manual, you can see that the reset value (default value) is 1 for some peripherals, meaning that they are powered on by default, whereas it is 0 (OFF by default) for others.

Example 1. Powering peripherals on

```
LPC_SC -> PCONP |= (1 << xx);
// where xx is the bit number in PCONP that controls the
// power (ON/OFF) for a specific peripheral.
```

The A/D converter (ADC) power is controlled by bit 12 of the `PCONP` register, which is 0 by default. *You must set that bit to power up your ADC.*

To save power, you can turn the power OFF for any unused peripherals that are ON by default.

3.1.2. Peripheral Clock

Most of the microcontroller peripherals, including timers and the ADC, require setting a peripheral clock (`PCLK`) to drive the peripheral.

There are four possible frequency configurations for the peripheral clock (`PCLK`), which are set using a pair of bits.

Table 1. Peripheral Clock (`PCLK`) Frequency Configurations

Bit Values	Frequency Configuration
`01`	`PCLK` = `CCLK`
`10`	`PCLK` = `CCLK` / 2
`00`	`PCLK` = `CCLK` / 4
`11`	`PCLK` = `CCLK` / 8

These pairs of bits belong to the `PCLKSEL0` and `PCLKSEL1` registers, which control the `PCLK` frequency for all peripherals.

The PCLKSEL0 and PCLKSEL1 Register Fields figure illustrates some of the fields of the PCLKSEL0 and PCLKSEL1 registers. Every two bits control the PCLK frequency for a specific peripheral.

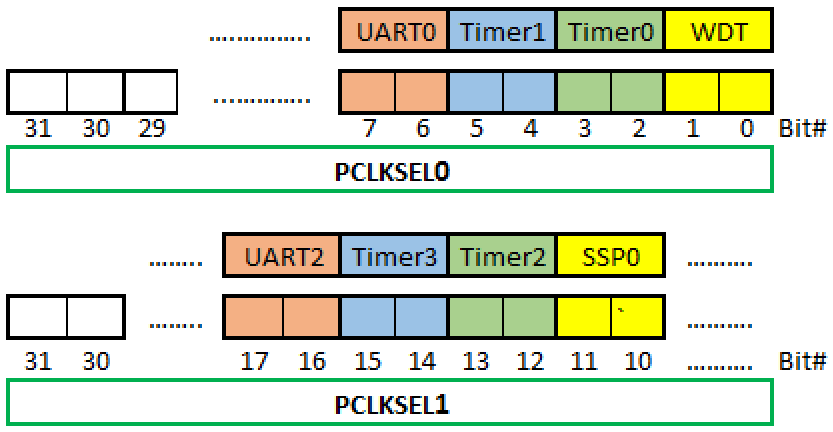


Figure 1. PCLKSEL0 and PCLKSEL1 Register Fields

Exercise

Refer to Chapter 4 in the LPC176x manual to find out the two bits needed to configure the PCLK frequency for the ADC.

Question

What would happen if you skip this step?

For the full list of peripherals and their corresponding two bits in PCLKSEL0 or PCLKSEL1, you can refer to Chapter 4 (section 4.7.3) in the LPC176x manual.

3.1.3. Pin Functions

3.2. The PINSELx Registers

This section is not specific to ADC. It is about configuring the function of a pin in a port. One possible functions is *ADC*.

Configuring the hardware involves a common step regardless of the hardware being configured. That common step is configuring the functions of the relevant pins.

Each pin can be configured to perform one of four functions. Therefore, the function of each pin is controlled by two bits, as follows:

00	Primary (default) function, (GPIO)
01	First alternate function
10	Second alternate function
11	Third alternate function

As such, to configure the functions of the five 32-bit ports, ten function selection registers are required. They are named PINSEL0, PINSEL1, PINSEL2, …, PINSEL9. PINSEL0 controls the functions of the lower half of port 0 (P0.0 to P0.15), PINSEL1 controls the functions of pins P0.16 to P0.31, PINSEL2 controls the functions of pins P1.0 to P1.15, and so on.

For example, the two least significant bits in PINSEL0 control the function of pin P0.0 as follows:

00	GPIO
01	RD1: CAN1 receiver input
10	TXD3: Transmitter output for UART3
11	SDA1: I2C1 data input/output

(See Table 73 in the LPC1768 User Manual.)

All PINSELx registers are fields in the LPC_PINCON structure.

So, to configure P0.0 to function as TXD3 instead of GPIO:

```
LPC_PINCON -> PINSEL0 = 0x00000002;   // Assignments like this are not the best way,
                                      // unless you want to set the remaining pins to GPIO
```

To avoid affecting other pins, You may want to use bitwise operations to set and/or clear the required bits in PINSELx.

Using 00 for any pin sets its function to GPIO. The reset value for PINSELx registers is 0x00000000. That is why the default function for all I/O pins after a reset is GPIO.

You may want to refer back to this section whenever you want a pin to have a function other than GPIO.

3.3. ADC Pins

Many microcontroller pins can be configured to perform one of many functions. To use the ADC, you must set the function of an appropriate pin to be analog input (AD0.x in the manual).

Exercise

Refer to Chapter 8 of the LPC176x manual to determine:

1. which PINSELx register should be modified
2. which bits of the register should be modified
3. what value should the bits be set to

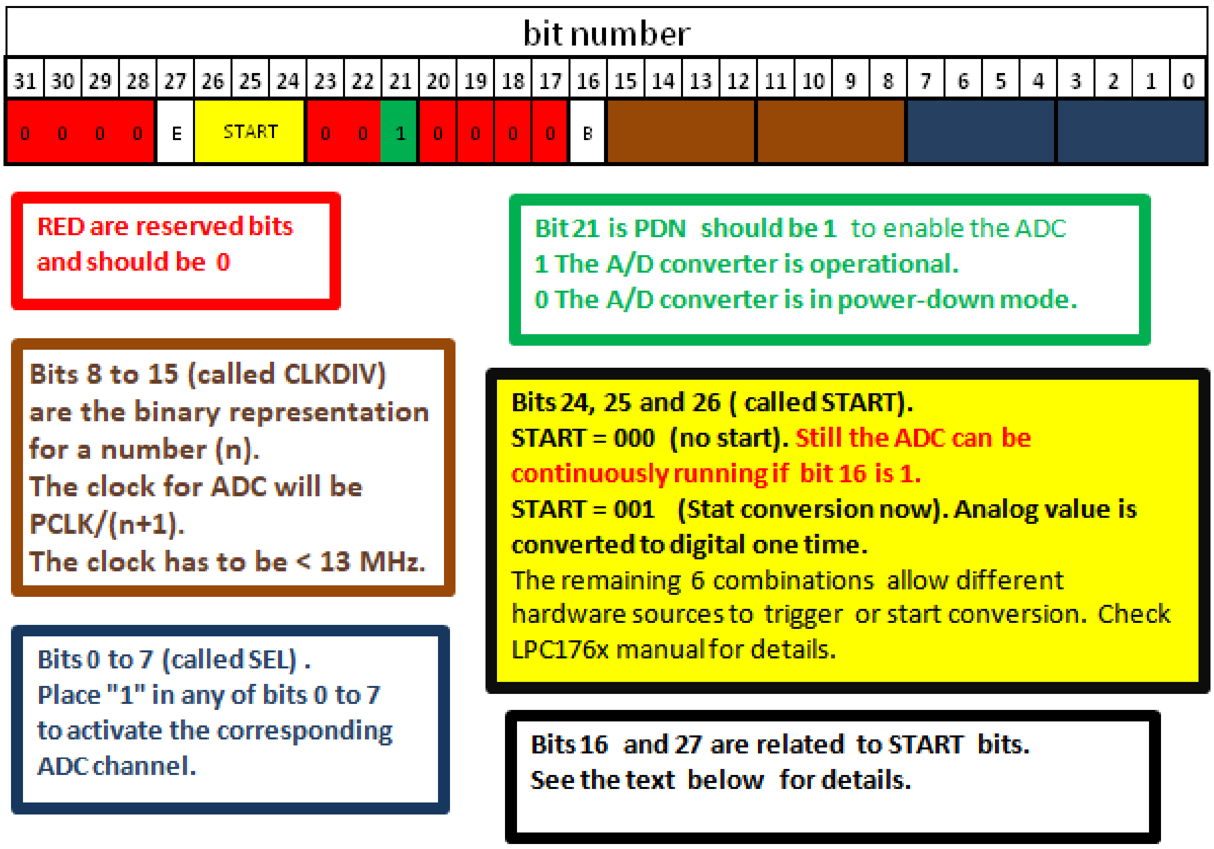
You should connect a device that generates an analog voltage signal to the selected pin. Examples of such devices are light sensors (LDR) and potentiometers.

It is professional to correctly address the above three issues for every peripheral you plan to use, regardless of the defaults.

3.4. ADC Configuration

The main setup register for the ADC is the *A/D Control Register* (`AD0CR`). The `AD0CR` Register Fields figure illustrates the fields of the `AD0CR` register.

There is only *one* ADC in the LPC1768 microcontroller. In the `LPC17xx.h` header file, the control register is referred to as `ADCR`; while in the chip manual it is called `AD0CR`. The reason for that is that some other chips have multiple ADCs, named: `AD0CR`, `AD1CR`, etc.



RED are reserved bits and should be 0

Bit 21 is PDN should be 1 to enable the ADC
1 The A/D converter is operational.
0 The A/D converter is in power-down mode.

Bits 8 to 15 (called CLKDIV) are the binary representation for a number (n).
The clock for ADC will be PCLK/(n+1).
The clock has to be < 13 MHz.

Bits 24, 25 and 26 (called START).
START = 000 (no start). Still the ADC can be continuously running if bit 16 is 1.
START = 001 (Stat conversion now). Analog value is converted to digital one time.
The remaining 6 combinations allow different hardware sources to trigger or start conversion. Check LPC176x manual for details.

Bits 0 to 7 (called SEL) .
Place "1" in any of bits 0 to 7 to activate the corresponding ADC channel.

Bits 16 and 27 are related to START bits.
See the text below for details.

Figure 2. A/D Control Register (`AD0CR`) Fields

The following table explains the function of the `B` (*Burst*) and `E` (*Edge*) bits of the `AD0CR` register.

Bit 27 (`E`) works only if `B` = 0 and `START` ≥ 2. When 2 ≤ `START` ≤ 7, the conversion starts when the state of a specific pin is changed. The `E` bit decides whether the ADC is triggered on the positive edge or the negative edge of that pin specified by `START`.

Bit	Label	Value	Effect
16	`B`	0	The `START` bits control when the ADC starts the conversion
		1	The ADC is continuously running (`START` should be `000`)
27	`E`	0	Start conversion on a falling edge
		1	Start conversion on a rising edge

3.4.1. START vs. BURST

Using START will perform the conversion only once.

If you want the analog value to be repeatedly converted, you have two options:

1. Set the B bit (Burst) of the AD0CR register to 1; or
2. Set the START bits to 001 repeatedly, i.e. in a loop. The analog value is read every time such a statement is executed.

3.4.2. Using ADC Interrupt

In simple ADC applications, you don't need interrupts. You can simply read the digitized value from the proper register whenever needed and take some action. However, in some applications, such as real time applications, you may need to interrupt the CPU to take an action *only when* the conversion is completed. To do that, you can use the ADGINTEN register.

See Table 534 in Chapter 29 of the LPC176x manual for details.

3.5. Reading Digital Values

There are 8 ADC channels, each corresponding to an analog pin. The digitized value corresponding to an input analog voltage is stored in 12 bits in one of the *A/D Data Registers*: ADDR0 to ADDR7, where each register corresponds to an analog pin.

The ADDR Register Fields figure illustrates the fields of the ADDRx registers.

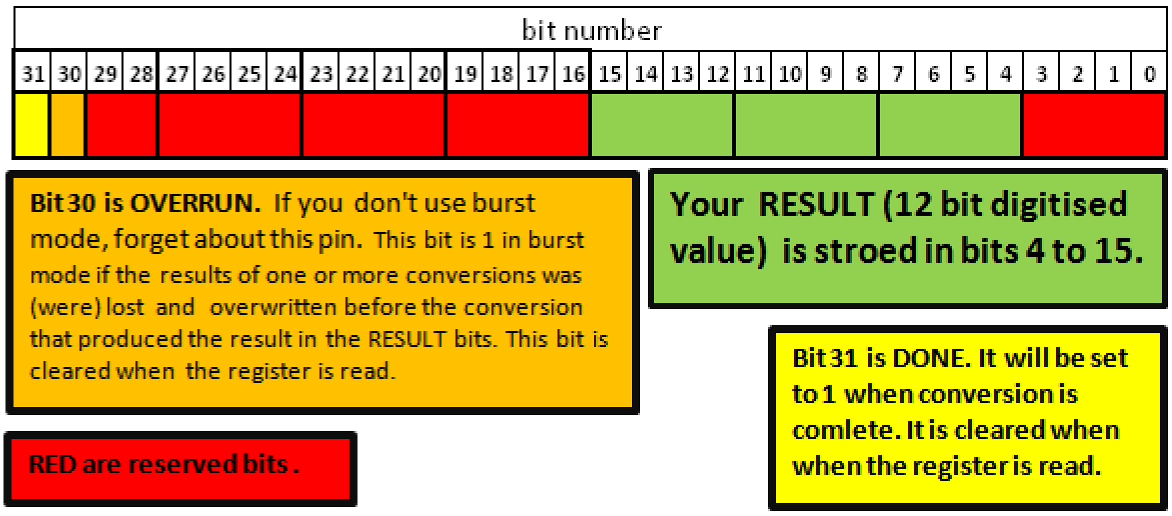

bit number
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bit 30 is OVERRUN. If you don't use burst mode, forget about this pin. This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits. This bit is cleared when the register is read.
Your RESULT (12 bit digitised value) is stroed in bits 4 to 15.
Bit 31 is DONE. It will be set to 1 when conversion is comlete. It is cleared when when the register is read.
RED are reserved bits.

Figure 3. A/D Data Register (ADDR) Fields

Using proper shifting and bitwise operations, you should be able to get the proper value representing the analog voltage.

The DONE and OVERRUN bits are less important (may not be needed) in BURST mode. However in START mode, you may need to check them to avoid reading an old or unintended value.

Example 2. Using the `DONE` Bit

To wait until the conversion of the ADC channel 3 is over, you may use:

```
while ((LPC_ADC->ADDR3 & (1 << 31)) == 0);    // Check the DONE bit for ADC channel #3
```

The 12-bit digital value generated by the ADC ranges from 0 to 4095. The way to process this value depends on your application.

You may want to divide this range to a number of sub-ranges, and assign different actions for each sub-range. In this case, you can use an if-else block.

In many applications, however, you will want to map this range to a another range using a mathematical formula. For example, if you are reading from an analog temperature sensor, you would want to map the 0-to-4095 range to the range of temperatures supported by the sensor, as specified in the sensor's data sheet. In most cases, a linear relationship is sufficient.

3.6. Analog Output

To write analog values to an analog output device, use the LPC1768's digital-to-analog converter (DAC) as follows:

1. Use `PINSELx` to configure P0.26 to function as analog output (`AOUT`).
2. Use the *D/A Converter Register* (`DACR`) to set the digital value to be converted to analog.

   Refer to Chapter 30 in the LPC176x manual for details.

4. Tasks

1. Use the ADC in LPC1768 to read an analog input device, such as the LDR (light sensor) or the potentiometer.

   The output can be any thing you want. The seven-segment display is a good option. You can simply display the analog level. If you use one seven-segment display, you have 10 different levels (0 to 9).

   It is recommended to use a formula to map the readings to sensible values, instead of using an if-else block.

2. Use the DAC in LPC1768 to output analog values to an analog device.

5. Grading Sheet

Task	Points
Analog Input	5
Analog Output	2
Discussion	3

Resources

- [lpc1768-manual]

  NXP Semiconductors. *UM10360 — LPC176x/5x User Manual*. Rev. 3.1. 4 April 2014. https://www.waveshare.com/w/upload/0/07/LPC176x5x_User_manual_EN.pdf